

# Setting up NGINX for PHP

From Development Server to Production-Ready Web Server

- What We're Building
  - Current Setup (Development)
  - New Setup (Production-Ready)
- Understanding the Architecture
  - PHP Built-in Server
  - NGINX + PHP-FPM
  - Linux (Ubuntu/Debian)
- Step 5: Create a Test PHP File
  - Create `info.php` in your web root
  - Create `test.php` for API testing
  - Advanced NGINX Configuration (Optional)

# What We're Building





## Current Setup (Development)

```
php -S localhost:8000
```

- ❌ Single-threaded
- ❌ Development only
- ❌ No static file optimization
- ❌ Limited concurrent users

## New Setup (Production-Ready)

NGINX + PHP-FPM (PHP FastCGI Process Manager)

-  Multi-threaded
-  Production ready
-  Optimized static file serving
-  Handles thousands of users

# Understanding the Architecture

## PHP Built-in Server

Browser → PHP -S → PHP Script

Single process handles everything

## NGINX + PHP-FPM

Browser → NGINX → PHP-FPM → PHP Script  
↓  
Static Files (direct)

NGINX handles static files, PHP-FPM processes PHP

## Step 1: Install PHP-FPM

### Windows

PHP-FPM is included with PHP 7.4+ on Windows

```
# Check if available  
php --version  
php-cgi --version
```

## macOS

```
# PHP-FPM comes with Homebrew PHP  
brew install php
```

```
# Or if you need to reinstall  
brew reinstall php
```

## Linux (Ubuntu/Debian)

```
sudo apt update  
sudo apt install php-fpm php-mysql
```



## Step 2: Configure PHP-FPM

### Start PHP-FPM Service

macOS:

```
brew services start php  
# or manually: php-fpm
```

`brew services` command shows the `brew services` that are running on your system.

```
smcho@m4 www> brew services  
Name  Status  User  File  
mysql  started smcho ~/Library/LaunchAgents/homebrew.mxcl.mysql.plist  
nginx  started smcho ~/Library/LaunchAgents/homebrew.mxcl.nginx.plist  
php    started smcho ~/Library/LaunchAgents/homebrew.mxcl.php.plist
```

## Linux:

```
sudo systemctl start php-fpm      # CentOS/RHEL
sudo systemctl start php8.1-fpm   # Ubuntu (adjust version)
sudo systemctl enable php8.1-fpm  # Auto-start on boot
```

## **Windows:**

PHP-FPM runs automatically with proper NGINX configuration

## Step 3: Basic NGINX Configuration for PHP

### Find Your NGINX Configuration File

- **Windows:** `C:\tools\nginx\conf\nginx.conf` or `C:\nginx\conf\nginx.conf`
- **macOS:** `/usr/local/etc/nginx/nginx.conf` (/opt/homebrew/ for Apple Silicon Mac)
- **Linux:** `/etc/nginx/nginx.conf`

## Create a Basic Configuration

Replace the `server` block in `nginx.conf` :

```
server {  
    listen 80;  
    server_name localhost;  
    root /var/www/html; # Adjust path for your system  
    index index.php index.html index.htm;  
  
    location / {  
        try_files $uri $uri/ =404;  
    }  
  
    location ~ \.php$ {  
        fastcgi_pass 127.0.0.1:9000; # PHP-FPM address  
        fastcgi_index index.php;  
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
        include fastcgi_params;  
    }  
}
```

## Server Block Overview

```
server {  
    listen 8080;  
    server_name localhost;  
    root /usr/local/var/www;  
    index index.php index.html;
```

- listen 8080: Port used for HTTP (macOS often avoids port 80)
- server\_name localhost: Domain this config applies to
  - Change this when you use your web domain name
- root: Directory to serve files from
  - Change this for your convenience
  - *Warning!* For Windows, the path should use '/', not '\\.
- index: Default files to serve in a directory

## Static File Handling

```
location / {  
    try_files $uri $uri/ =404;  
}
```

- Tries to serve the exact file or folder
- If not found, returns 404
- Ensures clean URL support (e.g., /about)

## *PHP File Handling*

```
location ~ /\.php$ {  
    fastcgi_pass 127.0.0.1:9000;  
    fastcgi_index index.php;  
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
    include fastcgi_params;  
}
```

- Matches .php files via regex
- Forwards request to PHP-FPM at 127.0.0.1:9000
- Passes script path and required FastCGI parameters



## How It Works

1. Browser requests <http://localhost:8080/index.php>
2. NGINX matches .php and forwards to PHP-FPM
3. PHP-FPM executes index.php and returns output
4. NGINX sends a response back to the browser

 Efficient, production-ready setup for PHP apps

## Step 4: Restart nginx

- **Windows:** `nginx.exe -s reload`
- **Mac:** `nginx -s reload`
- **Linux (Ubuntu):** `sudo systemctl restart nginx`

## Step 5: Create a Test PHP File

Create `info.php` in your web root

```
<?php  
phpinfo();  
?>
```

## Create `test.php` for API testing

```
<?php
header('Content-Type: application/json');

$response = [
    'message' => 'NGINX + PHP-FPM is working!',
    'server' => $_SERVER['SERVER_SOFTWARE'] ?? 'Unknown',
    'php_version' => PHP_VERSION,
    'timestamp' => date('Y-m-d H:i:s'),
    'method' => $_SERVER['REQUEST_METHOD'],
    'uri' => $_SERVER['REQUEST_URI']
];

echo json_encode($response, JSON_PRETTY_PRINT);
?>
```

- There is no `sendResponse()`, but the web browser can display the JSON.
- PHP is sending a valid response with `Content-Type: application/json`, and your browser is rendering it as raw text, not HTML.
  - The browser does not "execute" or "render" JSON like HTML.
  - It simply displays the JSON text in the browser window.
  - That happens even if you don't explicitly use `sendResponse()`, because the `echo json_encode(...)` line is outputting JSON content directly.

- PHP automatically generates this as a response.

```
HTTP/1.1 200 OK
Server: nginx/1.24.0
Content-Type: application/json
Content-Length: 226
Connection: keep-alive

{
  "message": "NGINX + PHP-FPM is working!",
  "server": "nginx/1.24.0",
  "php_version": "8.2.12",
  "timestamp": "2025-08-02 18:01:30",
  "method": "GET",
  "uri": "/test.php"
}
```

## Step 6: Start Services and Test

### 1. Start PHP-FPM (if not already running)

macOS:

```
brew services start php
```

Linux:

```
sudo systemctl start php8.1-fpm # Ubuntu  
sudo systemctl start php-fpm    # CentOS
```

## 2. Reload NGINX Configuration

```
nginx -s reload
```

## 3. Test Your Setup

- Visit: <http://localhost/info.php> (or :8080 on macOS)
- Visit: <http://localhost/test.php>

You should see the PHP info and the JSON response!



## Advanced NGINX Configuration (Optional)

### Security headers

```
add_header X-Frame-Options "SAMEORIGIN" always;  
add_header X-Content-Type-Options "nosniff" always;
```

- X-Frame-Options: SAMEORIGIN
  - Prevents your site from being embedded in an `iframe` by other domains
  - Mitigates clickjacking attacks
- X-Content-Type-Options: nosniff
  - Tells browsers not to “sniff” content types
  - Enforces correct Content-Type
  - Prevents MIME type confusion and XSS
- always
  - Ensures headers are sent on all responses (even 404, 500)

## Static files optimization

```
location ~* \.(css|js|png|jpg|jpeg|gif|co|svg)$ {  
    expires 1y;  
    add_header Cache-Control "public, immutable";  
}
```

- NGINX Cache Control for Static Assets
  - location ~\* .(...) \$ — Matches static file types (case-insensitive)
  - expires 1y — Sets Expires header to 1 year in the future
  - add\_header Cache-Control "public, immutable":
    - public: Allows caching by any cache (CDN, browser, etc.)
    - immutable: Tells the browser the file won't change, so skip revalidation
- Result
  - Faster page loads
  - Reduced server load
  - Ideal for versioned assets like app.abc123.js

## PHP processing

```
location ~ /\.php$ {  
    try_files $uri =404;  
    fastcgi_pass 127.0.0.1:9000;  
    fastcgi_index index.php;  
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
    include fastcgi_params;  
}
```

Certainly! Here's a one-page Marp slide that explains the location ~ .php\$ block:

## NGINX PHP Handler Block Explained

```
location ~ /\.php$ {  
    try_files $uri =404;  
    fastcgi_pass 127.0.0.1:9000;  
    fastcgi_index index.php;  
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
    include fastcgi_params;  
}
```

- Used to connect NGINX with PHP-FPM for executing PHP scripts
  - `location ~ .php$` — Matches all .php files
  - `try_files $uri =404` — Returns 404 if the file doesn't exist
  - `fastcgi_pass 127.0.0.1:9000` — Sends the request to PHP-FPM over FastCGI
  - `fastcgi_index index.php` — Default file if directory is requested
  - `fastcgi_param SCRIPT_FILENAME ...` — Tells PHP-FPM the script's full path
  - `include fastcgi_params` — Loads required FastCGI variables (e.g., method, URI)

## API routing

```
location /api/ {  
    try_files $uri $uri/ /api/index.php?$query_string;  
}
```



- Common in PHP frameworks (e.g., Slim, Laravel) for handling API routes via a single front controller.
  - location /api/ — Matches any request starting with /api/
  - try\_files \$uri \$uri/ ... — Tries to serve a real file or folder
  - /api/index.php?\$query\_string — Fallback if not found:
  - Internally rewrites the request to index.php
  - Appends original query string (\$query\_string)

## Deny access to hidden files

```
location ~ /\. {  
    deny all;  
}
```

